Université Jean Monnet

Master 2, DSC 2025-2026
Deep Learning
Projet

Détection de falsification d'images scientifiques

Élèves :
Mohamed MOUDJAHED
Ozan GUNES
Tolga YILDI
Aymane ZENNOUHI

Professeur :
Amaury Habrard
Hadi El Zein

8 février 2026

Table des matières

Résumé

Ce travail porte sur la détection et la segmentation automatique des falsifications par copie-déplacement dans des images biomédicales scientifiques à l'aide de méthodes de deep learning. L'objectif est de localiser précisément, au niveau du pixel, les régions artificiellement dupliquées afin de contribuer à la lutte contre la manipulation frauduleuse des données visuelles en recherche. Les expériences sont menées sur un jeu de données composé de plusieurs centaines de falsifications confirmées, issues de plus de 2 000 articles scientifiques rétractés [1]. Différentes architectures de segmentation ont été fine-tunées et comparées, intégrant des backbones modernes tels que ResNet [3], SegFormer [12], CMSeg-Net [9], ConvNeXt [5], SAM2 [7], DINOv2 [6] et DINOv3 [10]. Un pipeline complet couvrant l'entraînement, la validation et le test a été mis en place, avec un suivi rigoureux des performances. L'évaluation repose sur les métriques oF1, Dice et IoU. Les résultats mettent en évidence l'influence du choix de l'architecture et du pré-entraînement sur les performances, et soulignent le potentiel des modèles récents fondés sur les transformeurs et l'apprentissage auto-supervisé pour la détection de manipulations d'images scientifiques. Les résultats obtenus montrent que le choix de l'architecture comme celui de la stratégie de pré-entraînement jouent un rôle déterminant dans les performances finales. Ils confirment aussi le potentiel prometteur des modèles fondés sur les transformeurs et sur l'apprentissage auto-supervisé pour ce type de tâche.

1 Introduction

La falsification d'images scientifiques, et plus particulièrement les manipulations par copie-déplacement, représente un enjeu majeur pour la crédibilité de la recherche biomédicale. La détection automatique de ces falsifications nécessite des méthodes capables de localiser avec précision les régions dupliquées au niveau du pixel, afin de fournir un outil fiable pour la lutte contre la fraude scientifique.

Dans ce travail, nous avons mis en place un pipeline complet couvrant toutes les étapes du processus de deep learning, incluant l'entraînement, la validation et le test des modèles. La qualité des prédictions a été évaluée à l'aide de métriques adaptées à la segmentation et à la détection de falsifications, telles que l'Of1 (implémentation reprise de Kaggle), le forgedOf1, le Dice, l'Intersection over Union (IoU), ainsi que le rappel et la précision. Ces métriques permettent de quantifier à la fois la précision de la localisation et la capacité à détecter toutes les régions manipulées.

Pour optimiser les performances, nous avons procédé à une analyse détaillée des résultats obtenus et réalisé des études d'ablation afin d'ajuster les architectures et les stratégies d'entraînement en fonction des observations. Plusieurs architectures modernes ont été testées et comparées, incluant des réseaux convolutifs classiques et des modèles basés sur les transformeurs, afin d'identifier la combinaison offrant les meilleures performances pour cette tâche spécifique.

2 Pipeline

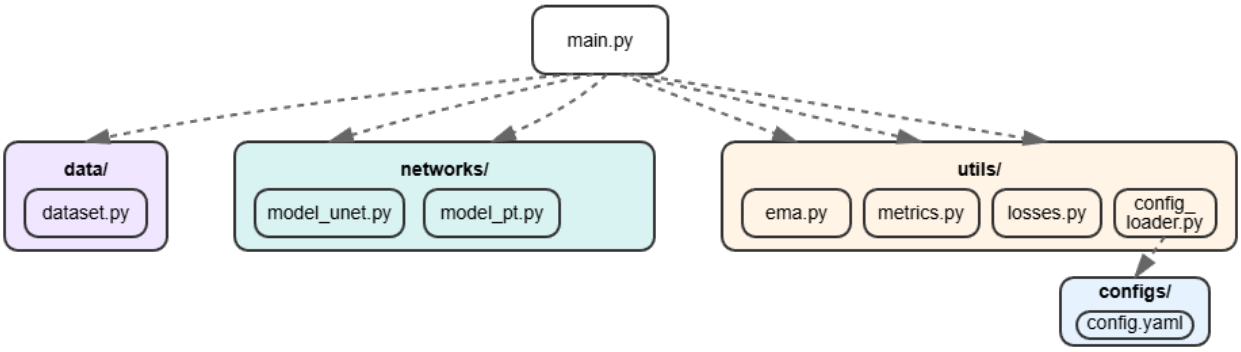

FIGURE 1 – Workflow d'entraînement, validation et test des modèles de détection de falsifications

2.1 Split des données et configuration commune

Split des données : Le jeu de données a été réparti en ensembles d'entraînement, de validation et de test de manière à garantir une diversité des niveaux de complexité dans chaque sous-ensemble.
L'ensemble d'entraînement contient 3619 échantillons (1956 forgés, 1663 authentiques) et l'ensemble de validation comprend 777 échantillons (420 forgés, 357 authentiques). La structure des données est organisée comme suit : `images/authentic`, `images/forged`, `masks/` (format `.npy`). Les masques sont disponibles uniquement pour les images forgées (1956 pour l'entraînement, 420 pour la validation).
Prétraitement : Les images sont redimensionnées à 518×518 ou 512×512 pixels selon le modèle utilisé. Ce choix a été effectué afin d'assurer un bon compromis entre performances et coût computationnel. Les images sont conservées dans l'intervalle [0, 255], puis converties en tenseurs RGB de dimension (C, H, W). Les instances sont extraites par analyse des composantes connexes à l'aide de `scipy.ndimage.label` lors de l'inférence.
Augmentation (train uniquement) :

- Type 0 : Aucune
- Type 1 : Augmentations photométriques et bruit (flou gaussien, flou de mouvement, flou médian, bruit gaussien, bruit ISO, ajustement de luminosité et contraste, correction gamma, CLAHE)
- Type 2 : Transformations géométriques et bruit (flips horizontal/vertical, rotation aléatoire à 90°, combinés aux augmentations du Type 1)

Taille de batch et normalisation : Tous les modèles ont été entraînés avec un batch effectif de 16, obtenu à partir d'un batch réel de 8 combiné à une accumulation de gradients sur 2 itérations. Ce choix nous a permis de stabiliser l'apprentissage tout en respectant nos contraintes matérielles. Pour les architectures de type U-Net, la normalisation par groupes (*Group Normalization*) a été privilégiée, la normalisation par batch étant peu adaptée aux faibles tailles de batch et susceptible d'entraîner une instabilité de l'entraînement.

Choix des hyperparamètres (baselines) : Concernant les modèles de référence (baselines), les hyperparamètres ont été choisis conformément aux recommandations issues de la documentation officielle, des dépôts GitHub correspondants, ou des configurations par défaut proposées par la bibliothèque HuggingFace.
Gel des backbones et stratégie d'apprentissage : Pour l'ensemble des modèles, les poids des backbones ont été gelés durant les cinq premières époques afin de favoriser une adaptation progressive des têtes de segmentation. Une exception a été faite pour le modèle DinoV3+UPerNet, pour lequel le gel a été maintenu pendant dix époques, ce dernier présentant une instabilité marquée après cinq époques d'entraînement.Un multiplicateur du taux d'apprentissage spécifique au backbone (`backbone_lr_multiplier = 0.1`) a également été appliqué, permettant d'ajuster la vitesse d'apprentissage entre le backbone pré-entraîné et les couches nouvellement ajoutées.
Optimisation et régularisation : Afin de limiter les phénomènes d'explosion du gradient observés lors des premières phases d'expérimentation, un mécanisme de *gradient clipping* a été systématiquement employé. Cette stratégie a contribué à améliorer significativement la stabilité de l'entraînement.
L'ordonnancement du taux d'apprentissage (*learning rate scheduler*) a été combiné à un mécanisme d'arrêt anticipé (*early stopping*) basé sur la maximisation du score oF1 sur l'ensemble de validation, avec une patience fixée à 25 époques.
Fonctions de perte et implémentation : Les fonctions de perte évaluées au cours de nos expérimentations reposent sur des combinaisons pondérées de la Binary Cross-Entropy (BCE) avec la Dice Loss ou la Focal Loss. Dans les deux cas, une pondération équilibrée de 50%–50% a été adoptée afin de concilier précision locale et robustesse face au déséquilibre des classes.
L'ensemble des implémentations a été réalisé à l'aide du framework PyTorch sur Python.

3 Architectures d'entrainement

SAM2+U-Net

SAM2 (Segment Anything Model 2) est un modèle de segmentation développé par Meta, reposant sur un encodeur hiérarchique nommé Hiera [8], conçu pour extraire des représentations visuelles multi-échelles. Le traitement débute par une division de l'image d'entrée en patches de 4×4 pixels, qui sont ensuite analysés par une succession de blocs Transformers [11] organisés en quatre niveaux. Chaque niveau applique des mécanismes d'attention et de pooling progressifs, permettant de réduire progressivement la résolution spatiale tout en augmentant la richesse des caractéristiques extraites. Ce processus aboutit à la production de quatre niveaux de représentation, correspondant à des résolutions de 1/4, 1/8, 1/16 et 1/32 de l'image originale.

Cette architecture hiérarchique permet à SAM2 de capturer simultanément des détails locaux précis et des structures sémantiques globales. Un module de type FPN (Feature Pyramid Network) [4] est ensuite utilisé pour harmoniser ces différents niveaux en normalisant leurs dimensions à 256 canaux. Par ailleurs, SAM2 a été pré-entraîné sur un vaste ensemble de données composé de millions d'images, ce qui lui donne une forte capacité à identifier et segmenter des objets de nature variée.

ConvNeXt+U-Net

ConvNeXt [5] est une architecture convolutionnelle moderne développée par Meta AI qui modernise les réseaux ResNet en incorporant les innovations des Vision Transformers tout en conservant la simplicité des CNN. Le modèle utilise un backbone ConvNeXt-Base pré-entraîné sur ImageNet-22k (14M d'images) puis fine-tuné sur ImageNet-1k.

L'architecture repose sur un encodeur hiérarchique à 4 stages produisant des feature maps à résolutions décroissantes (1/4, 1/8, 1/16, 1/32 de l'image originale), avec respectivement 128, 256, 512 et 1024 canaux pour ConvNeXt-Base. Ces représentations multi-échelles sont ensuite traitées par un décodeur U-Net pour reconstruire la carte de segmentation finale.

DinoV2+U-Net

DINOv2 [6] est un modèle développé par Meta AI qui utilise une architecture Vision Transformer (ViT) [11]. L'idée principale derrière DINOv2 est d'apprendre des représentations visuelles de manière auto-supervisée, c'est-à-dire sans avoir besoin d'annotations manuelles. Le modèle commence par découper l'image d'entrée en petits patches de 14×14 pixels, qui sont ensuite transformés en vecteurs (embeddings) grâce à une couche de projection. Ces embeddings passent ensuite à travers plusieurs blocs Transformers qui utilisent des mécanismes d'attention multi-têtes pour capturer les relations entre les différentes parties de l'image.
Contrairement à des architectures comme ResNet [**?**] qui réduisent progressivement la résolution, DINOv2 garde la même résolution spatiale tout au long du réseau. Cependant, les représentations deviennent de plus en plus abstraites à mesure qu'on avance dans les couches. Pour notre tâche de segmentation, on extrait les caractéristiques à différentes profondeurs du Transformer (aux couches 6, 12, 18 et 24 pour un ViT-Large). Cela nous permet de récupérer à la fois des détails fins de texture et des informations sémantiques plus globales. Ces représentations sont ensuite réorganisées pour former des cartes de caractéristiques 2D compatibles avec un décodeur.

Un avantage majeur de DINOv2 est qu'il a été pré-entraîné sur un énorme dataset de 142 millions d'images (LVD-142M). Ce pré-entraînement auto-supervisé lui donne une excellente capacité de généralisation, ce qui est particulièrement intéressant pour notre cas d'usage sur des images scientifiques.

DinoV3+U-Net

DINOv3 [10] est la troisième génération du modèle DINO, publiée par Meta AI en août 2025. Cette version a été pré-entraînée sur un dataset de 1,7 milliard d'images (LVD-1689M), soit 12 fois plus que les 142 millions d'images de DINOv2. Nous utilisons la version ViT-Base distillée, qui conserve l'essentiel des performances du modèle enseignant tout en restant adaptée à des contraintes de calcul raisonnables.

L'innovation principale de DINOv3 est une technique appelée *Gram Anchoring*. Les chercheurs ont observé que lors d'un entraînement prolongé des modèles auto-supervisés, les performances sur les tâches globales (comme la classification) continuent de s'améliorer, mais les caractéristiques denses au niveau des patches se dégradent progressivement. Ce phénomène, où les patches perdent leur spécificité locale et deviennent trop similaires au token CLS global, est

problématique pour la segmentation. Le Gram Anchoring résout ce problème en utilisant des checkpoints de début d'entraînement comme "enseignants Gram" et en introduisant une fonction de perte qui force la matrice de Gram des caractéristiques actuelles à rester alignée avec celle de ces enseignants, préservant ainsi la qualité des caractéristiques denses.

DINOv3 apporte également des améliorations architecturales notables. Les embeddings positionnels appris de DINOv2 sont remplacés par des embeddings RoPE (*Rotary Position Embedding*), qui offrent une meilleure stabilité à travers différentes résolutions et ratios d'aspect. De plus, une phase de fine-tuning haute résolution permet au modèle de produire des cartes de caractéristiques stables même sur des images de très haute résolution.

CM-SegNet

CMSeg-Net (Copy-Move Segmentation Network) est un modèle de segmentation développé pour la détection de falsifications par copier-coller dans les images biomédicales, reposant sur une architecture encodeur-décodeur multi-résolution. Le modèle utilise MobileNetv2, pré-entraîné sur ImageNet. MobileNetv2 génère des features à différentes échelles de résolution qui sont ensuite traitées par des modules d'attention spécialisés.

Le cœur de l'architecture repose sur le module CoSA (Correlation-assisted Spatial Attention), conçu pour détecter les similarités intra-image caractéristiques des régions dupliquées. Ce module se compose de deux sous-modules complémentaires, le sous-module CoR (Correlation) qui calcule une matrice d'affinité entre les features pour identifier les régions présentant une forte corrélation spatiale, et le sous-module VRSA (Variable-sized Receptive field Spatial Attention) qui combine un bloc ASPP (Atrous Spatial Pyramid Pooling) avec un mécanisme d'attention spatiale pour capturer des informations contextuelles à différentes échelles de réception. Les modules CoSA sont appliqués aux tenseurs de features intermédiaires extraits par MobileNetv2, permettant d'identifier les duplications à plusieurs niveaux de résolution.

Le décodeur utilise des blocs résiduels inversés (Inverted Residual Blocks) pour reconstruire progressivement la carte de segmentation finale, en combinant les features enrichies par les modules CoSA avec les features multi-échelles de l'encodeur.

ResNet50

ResNet-50 (Residual Network à 50 couches) est un modèle de classification développé par Microsoft Research [3], reposant sur un encodeur à connexions résiduelles conçu pour extraire des représentations visuelles multi-échelles. Le traitement débute par une convolution initiale de 7×7 pixels suivie d'un max-pooling, puis l'image est analysée par une succession de blocs résiduels organisés en quatre niveaux. Chaque niveau applique des convolutions progressives qui réduisent la résolution spatiale tout en augmentant la profondeur des caractéristiques extraites, produisant quatre niveaux de représentation à des résolutions de 1/4, 1/8, 1/16 et 1/32 de l'image originale, avec respectivement 256, 512, 1024 et 2048 canaux.

Cette architecture résiduelle permet à ResNet-50 de capturer simultanément des détails locaux et des structures sémantiques globales, tout en résolvant le problème de dégradation du gradient grâce aux connexions de raccourci [3]. Par ailleurs, ResNet-50 a été pré-entraîné sur ImageNet

[2], composé de plus d'un million d'images, ce qui lui confère une capacité étendue à identifier des caractéristiques visuelles de nature variée.

U-Net et UperNet

U-Net est un réseau encodeur-décodeur utilisé comme décodeur de segmentation, combiné à différents backbones pour exploiter des représentations multi-échelles. Ses connexions de saut permettent de fusionner les informations locales et globales. UperNet est une architecture plus récente et polyvalente, intégrant également des backbones CNN ou Transformer, ainsi que des modules pyramidaux multi-échelles, ce qui lui permet de segmenter des objets de tailles variées et des scènes complexes.

4 Études d'Ablation et Configurations d'Entraînement

Nous avons mené des études d'ablation pour analyser l'impact des choix de conception et d'hyperparamètres sur nos modèles. Pour chaque architecture, une configuration de référence a été établie à partir des hyperparamètres recommandés dans les publications originales.
Nous avons ensuite exploré différentes valeurs de weight decay et learning rate, ainsi que plusieurs fonctions de perte, afin d'optimiser les performances et limiter le sur-apprentissage. Enfin, pour l'inférence, une recherche en grille a été effectuée sur les paramètres de post-traitement (min_area : {50, 100}, threshold : {0.25, 0.5, 0.75}) afin de sélectionner la combinaison optimale pour chaque modèle.

Au total, nos ablations ont permis de tester toutes ces configurations pour les différents backbones et décodeurs, comme résumé dans le tableau ci-dessous.

TABLE 1 – Configurations d'entraînement des modèles de détection de falsification

Modèle	Augmentation	Loss	Learning Rate	Batch Size	Weight Decay	Image Size	Backbone
dinov2_unet	0	BCE+DICE	0.0001	16	0.001	518	base
dinov2_unet	1	BCE+DICE	0.0001	16	0.001	518	base
dinov2_unet	2	BCE+DICE	0.0001	16	0.001	518	base
dinov2_unet	2	BCE+DICE	0.0003	16	0.01	518	base
dinov2_unet	2	BCE+Focal	0.0001	16	0.01	518	base
resnet50_unet	0	BCE+DICE	0.0001	16	0.001	512	ResNet50
resnet50_unet	1	BCE+DICE	0.0001	16	0.001	512	ResNet50
resnet50_unet	2	BCE+DICE	0.0001	16	0.001	512	ResNet50
resnet50_unet	2	BCE+DICE	0.0001	16	0.0005	512	ResNet50
segformer_b5	2	BCE+DICE	0.0001	16	0.001	512	B5
segformer_b3	2	BCE+DICE	0.0001	16	0.001	512	B3
cmsegnet	0	BCE+DICE	0.0001	16	0.001	512	base
cmsegnet	2	BCE+DICE	0.0001	16	0.001	512	base
sam2_unet	2	BCE+DICE	0.0003	16	0.001	512	base
sam2_unet	1	BCE+DICE	0.0003	16	0.001	512	base
sam2_unet	0	BCE+DICE	0.0003	16	0.001	512	base
dinov3_unet	0	BCE+DICE	0.0001	16	0.001	518	base
dinov3_unet	1	BCE+DICE	0.0001	16	0.001	518	base
dinov3_unet	2	BCE+DICE	0.0001	16	0.001	518	base
dinov3_unet	2	BCE+DICE	0.0001	16	0.01	518	base
dinov3_upernet	2	BCE+DICE	0.0001	16	0.001	518	base

5 Résultats

5.1 Vue d'ensemble multi-métrique

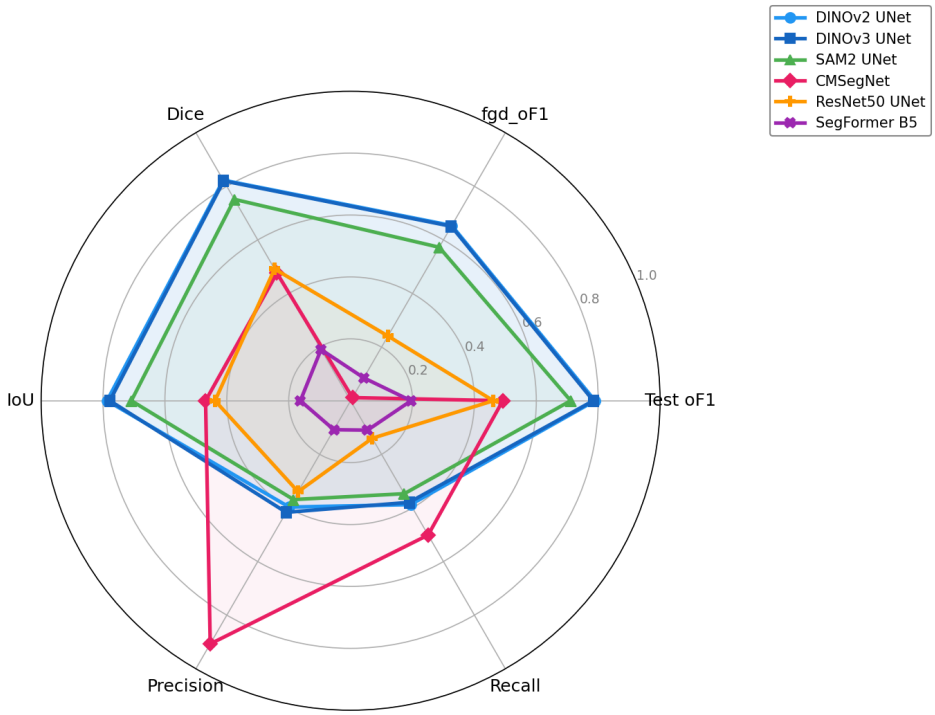

FIGURE 2 – Diagramme radar comparatif des six modèles sur l'ensemble des métriques.

Le diagramme radar fait apparaître une stratification nette des architectures en trois niveaux de performance. DINOv2 domine avec les meilleurs scores globaux (oF1 ≈ 0.79, Dice ≈ 0.82), suivi de près par DINOv3 (oF1 ≈ 0.78, Dice ≈ 0.82), puis SAM2 (oF1 ≈ 0.71, Dice ≈ 0.75). Les modèles restants, ResNet50, SegFormer et CMSegNet, affichent des performances nettement inférieures.

La supériorité des encodeurs DINO s'explique par leur pré-entraînement auto-supervisé sur de vastes corpus d'images, qui leur confère des représentations hiérarchiques capables de capturer aussi bien les micro-artefacts locaux que les incohérences structurelles globales — deux signatures typiques des falsifications. Le fait que DINOv2 surpasse légèrement DINOv3 est notable : malgré un corpus d'entraînement plus vaste et des innovations architecturales supplémentaires, DINOv3 ne parvient pas à creuser l'écart, ce qui suggère un possible effet de saturation des représentations pour cette tâche spécifique. SAM2, conçu pour la segmentation d'objets, dispose de représentations visuelles solides mais non spécifiquement adaptées à la détection d'anomalies, ce qui explique son retrait d'environ 8 points d'oF1.

Le cas de CMSegNet est paradoxal. Bien qu'explicitement conçu pour la détection de *copy-move*, il affiche une précision élevée ($P \approx 0.91$) mais un fgd_oF1 quasi nul (≈ 0.02) et un rappel limité ($R \approx 0.50$). Ce résultat contre-intuitif suggère que ses modules de corrélation, bien que capables d'identifier certaines duplications avec certitude, peinent à généraliser face à notre jeu de données.

ResNet50 et SegFormer échouent sur l'ensemble des indicateurs. Le premier souffre d'un champ réceptif trop local pour relier des régions distantes ; le second, pré-entraîné sur des scènes naturelles, subit un décalage de domaine trop important pour que ses représentations soient transférables à la détection de falsifications.

Il convient de nuancer ces résultats : les scores de précision et de rappel restent globalement faibles pour tous les modèles (P et R rarement au-dessus de 0.42), ce qui indique que même les meilleures architectures peinent à localiser finement les régions falsifiées. La tâche demeure intrinsèquement difficile.

5.2 Analyse des dynamiques d'apprentissage et de généralisation

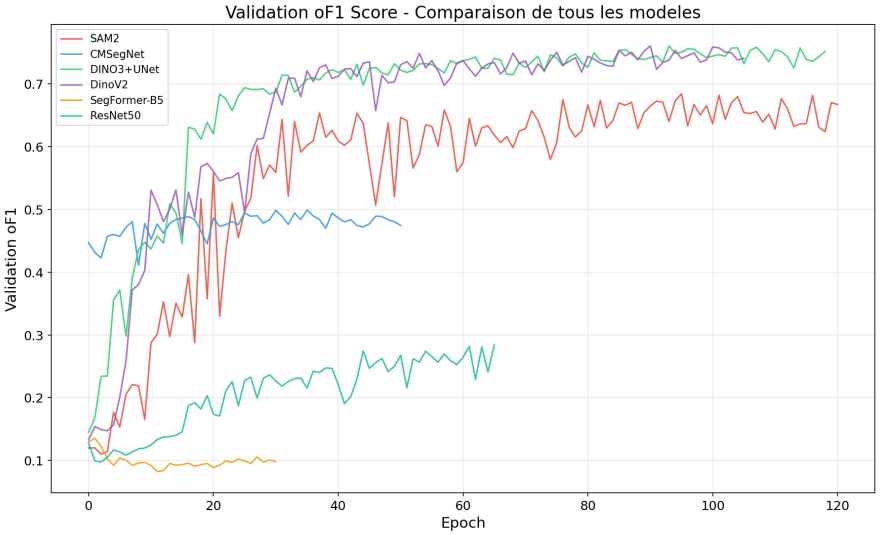

FIGURE 3 – Évolution du score oF1 de validation au cours de l'entraînement.

FIGURE 4 – Écart entre loss de validation et loss d'entraînement au fil des époques.

L'analyse conjointe des courbes d'apprentissage et de l'écart entre pertes permet d'évaluer à la fois la vitesse de convergence et la capacité de généralisation des différents modèles.

Les modèles auto-supervisés de type vision transformer, DINOv3 et DINOv2, présentent les dynamiques les plus favorables. DINOv3 converge rapidement vers un score oF1 élevé (0.70–0.75) dès les premières dizaines d'époques, tout en maintenant un faible écart entre les pertes d'entraînement et de validation (< 0.15). Ce comportement indique un transfert efficace de leurs représentations pré-entraînées, nécessitant peu d'adaptation pour la tâche de détection. DINOv2 suit une trajectoire similaire, avec une convergence légèrement plus lente et des performances marginalement inférieures (0.65–0.70), associées à un surapprentissage modéré (0.2–0.4), probablement lié à un pré-entraînement à plus petite échelle.

À l'inverse, SAM2 présente une dynamique instable, caractérisée par de fortes oscillations du score oF1 et une convergence tardive autour de 0.60–0.65. Cette instabilité s'accompagne d'un surapprentissage marqué, avec un écart de pertes atteignant 0.4–0.6. Elle reflète la difficulté d'adapter des représentations initialement conçues pour la segmentation d'objets vers la détection de manipulations, nécessitant une réorganisation profonde des features internes.

CMSegNet atteint rapidement un plateau autour de 0.48–0.50, sans amélioration ultérieure. Son écart de pertes reste faible, non pas en raison d'une bonne généralisation, mais du fait d'un apprentissage prématurément saturé. Les modules de corrélation détectent efficacement certaines duplications, mais ne parviennent pas à capturer la diversité des artefacts présents dans les données.

Les modèles SegFormer et ResNet50, donnent de moins bons résultats. SegFormer reste autour de 0,10, ce qui montre qu'il n'apprend pas assez. Il est surtout conçu pour la segmentation d'images générales et n'est pas adapté à l'analyse fine des traces de manipulation. ResNet50 fait un peu mieux (entre 0,25 et 0,30), mais ses performances restent limitées. Son champ de vision est trop restreint, ce qui l'empêche de bien détecter les incohérences à l'échelle globale.

5.3 Évaluation détaillée : oF1 global, zones falsifiées et compromis précision–rappel

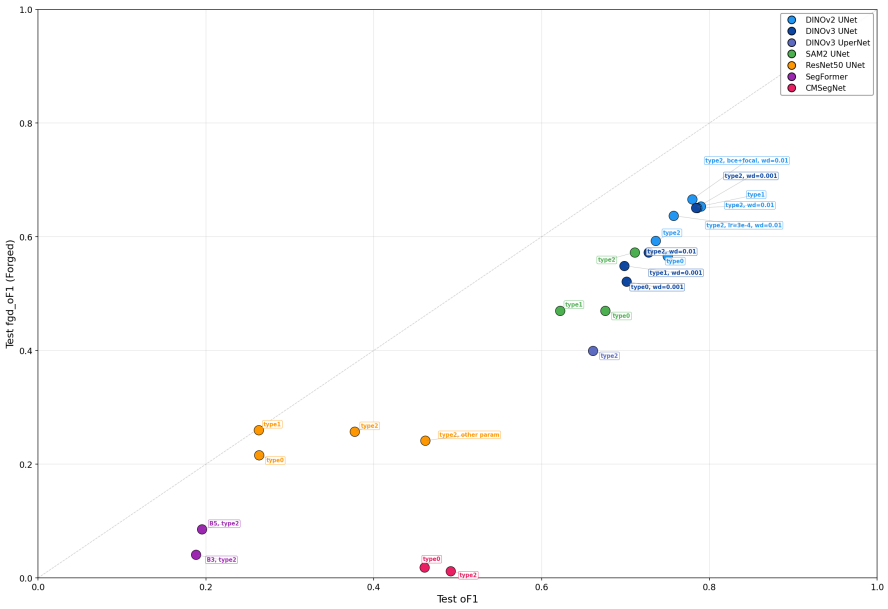

FIGURE 5 – Relation entre le score oF1 global et le fgd_oF1 (zones falsifiées uniquement).

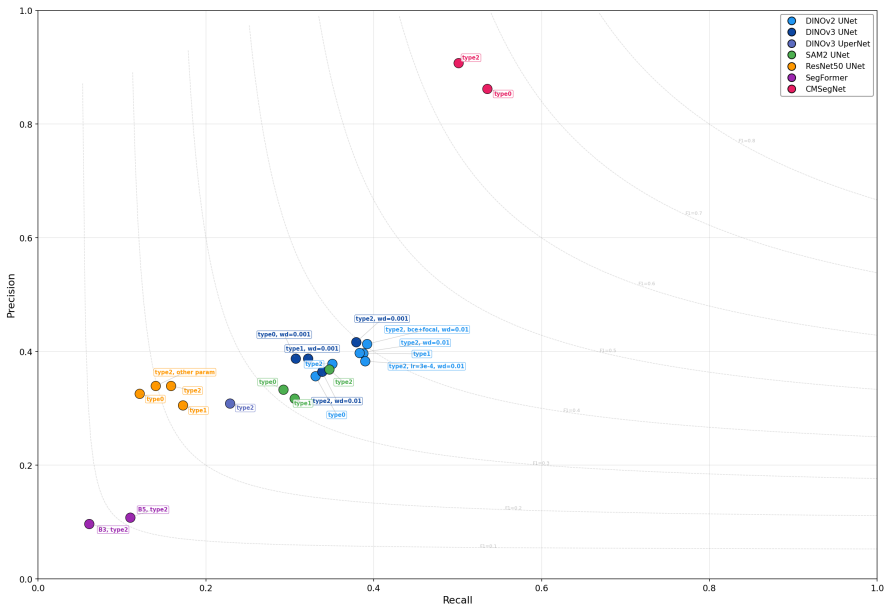

FIGURE 6 – Distribution précision/rappel pour toutes les configurations, avec iso-courbes F1.

La comparaison entre le score oF1 global et le fgd_oF1 (calculé uniquement sur les zones falsifiées) montre que la détection précise des régions manipulées reste plus difficile que ne le suggère le score global. La classe « authentique » étant majoritaire, un modèle peut obtenir un oF1 global correct en classant correctement les pixels non-falsifiés, tout en échouant sur les zones falsifiées.

DINOv3 se distingue par des performances élevées à la fois globalement (oF1 = 0.70–0.79) et sur les zones falsifiées (fgd_oF1 = 0.50–0.67), confirmant sa supériorité même sur les zones complexes. Pour DINOv2, la configuration **BCE+DICE avec weight decay 0.01** surpasse celle à 0.001, offrant un meilleur équilibre entre oF1 global et fgd_oF1. L'utilisation de la **focal loss** en complément de BCE+DICE améliore encore le fgd_oF1, en pondérant davantage les pixels falsifiés et en compensant le déséquilibre des classes.
CMSegNet illustre les limites structurelles de certains modèles : son oF1 global reste correct ($\approx$ 0.50), mais son fgd_oF1 est quasi nul (0.02). Les modules de corrélation ne détectent que les duplications intra-image très marquées et échouent sur la majorité des falsifications. ResNet50 et SegFormer obtiennent de faibles scores sur ces deux métriques, confirmant que leurs architectures ne conviennent pas à l'identification fine des artefacts.
L'analyse du compromis précision–rappel (Figure 6) confirme ces observations. Trois groupes se distinguent :

— **CMSegNet** : précision très élevée (0.85–0.90) mais rappel très faible (0.10–0.15), illustrant un comportement conservateur qui évite les faux positifs mais ignore la majorité des falsifications.

— **DINOv2, DINOv3 et SAM2** : précision et rappel intermédiaires ($\approx$ 0.30–0.45), traduisant un équilibre entre couverture des anomalies et gestion des fausses alertes. DINOv2 avec BCE+DICE (weight decay 0.01) ou focal loss atteint les meilleurs résultats pour le fgd_oF1. SAM2 montre un léger avantage en rappel, cohérent avec sa tendance à segmenter davantage de régions.

— **ResNet50 et SegFormer** : scores faibles sur les deux métriques, confirmant l'inadaptation de leurs architectures à cette tâche.

Aucun modèle n'atteint d'iso-courbe F1 supérieure à 0.6, soulignant la difficulté intrinsèque de la détection de falsifications et la marge importante de progression encore existante.

Modèle	Type	oF1	Dice	IoU	P	R
DinoV2-UNet	0	75.01	78.02	74.41	35.65	33.05
	1	78.55	81.84	78.13	39.77	38.29
	2	**78.97**	**82.30**	**78.71**	39.68	**38.76**
ResNet50-UNet	0	26.33	27.64	22.70	**32.60**	12.08
	1	26.29	28.62	22.69	30.55	**17.23**
	2	**46.13**	**49.16**	**43.81**	33.94	14.00
SAM2-UNet	0	67.60	71.85	67.57	33.28	29.22
	1	62.22	66.78	62.17	31.74	30.57
	2	**71.11**	**75.14**	**70.84**	**36.87**	**34.69**
DinoV3-UNet	0	70.14	74.54	69.66	38.76	30.71
	1	69.84	74.12	69.45	38.74	32.18
	2	**78.42**	**82.03**	**77.79**	**41.64**	**37.90**
ConvNeXt-UNet	0	50.15	17.69	12.85	23.48	15.75
	1	49.09	18.17	13.25	20.52	18.14
	2	**52.08**	**20.73**	**15.43**	21.20	**22.65**

TABLE 2 – Comparaison détaillée par architecture et type d'augmentation

Modèle	Test oF1	Dice	IoU	Test fgd_oF1
DinoV2-UNet type2 (decay 0.01)	**78.97**	**82.30**	**78.71**	65.37
DinoV3-UNet type2 (wd 0.001)	78.42	82.03	77.79	**65.08**
DinoV2-UNet type2 (BCE+Focal)	77.96	81.23	77.07	66.60
SAM2-UNet type2	71.11	75.14	70.84	57.26
ConvNeXt-UNet type2	49.69	22.16	16.65	25.16
ResNet50-UNet type2 (other)	46.13	49.16	43.81	24.14

TABLE 3 – Comparaison des meilleurs modèles (Type 2)

5.4 Seuil de Binarisation et Taille Minimale des Régions

Ces ablations sont effectuées en post-processing lors de l'inférence, sans nécessiter de réentraînement.

Expérience	Th	Area	Val oF1	Test oF1	Test fgd_oF1	Dice	IoU	P	R	Prénom
DINOv2 UNet										
dinov2_unet_type0	0.75	50	0.7402	0.7501	0.5662	0.7802	0.7441	0.3565	0.3305	Ozan
dinov2_unet_type1	0.75	50	0.7806	**0.7897**	0.6537	**0.8230**	**0.7871**	0.3968	0.3876	Ozan
dinov2_unet_type2	0.75	50	0.7285	0.7361	0.5927	0.7717	0.7309	0.3787	0.3504	Ozan
dinov2_unet_forgery type2_decay0.01	0.75	50	**0.7884**	0.7855	0.6508	0.8184	0.7813	0.3977	0.3829	Ozan/Tolga
dinov2_unet_type2 _lr0.0003_wd0.01	0.75	50	0.7392	0.7575	0.6371	0.7939	0.7528	0.3833	0.3899	Mohamed
dinov2_unet_type2 _bce+focal_wd0.01	0.75	50	0.7631	0.7796	**0.6660**	0.8123	0.7707	0.4130	0.3919	Mohamed
ConvNeXt-Base UNet										
convnext_unet_type0	0.5	50	0.5015	0.4946	0.2253	0.1842	0.1333	0.2507	0.1608	Aymane
convnext_unet_type1	0.5	50	0.4909	0.4564	0.2271	0.1905	0.1399	0.2482	0.1680	Aymane
convnext_unet_type2	0.5	50	0.5208	0.4969	0.2516	0.2216	0.1665	0.2419	0.2243	Aymane
DINOv3 UNet / UperNet										
dinov3_unet type2_wd0.001	0.5	50	0.7693	0.7842	0.6508	0.8203	0.7779	0.4164	0.3790	Mohamed
dinov3_unet type1_wd0.001	0.5	50	0.7034	0.6984	0.5491	0.7412	0.6945	0.3874	0.3218	Mohamed
dinov3_unet type0_wd0.001	0.5	50	0.6998	0.7014	0.5214	0.7454	0.6966	0.3876	0.3071	Mohamed
dinov3_unet type2_wd0.01	0.5	50	0.7110	0.7277	0.5725	0.7625	0.7231	0.3643	0.3384	Mohamed
dinov3_upernet type2	0.5	50	0.6450	0.6613	0.3996	0.6919	0.6332	0.3089	0.2286	Mohamed
ResNet50 UNet										
resnet50_unet_type0	0.75	100	0.2667	0.2633	0.2156	0.2764	0.2270	0.3260	0.1208	Tolga
resnet50_unet_type1	0.75	100	0.2727	0.2629	0.2602	0.2862	0.2269	0.3055	0.1723	Tolga
resnet50_unet_type2	0.75	100	0.3638	0.3771	0.2572	0.3965	0.3383	0.3397	0.1581	Tolga
resnet50_unet type2	0.75	100	0.4338	0.4613	0.2414	0.4916	0.4381	0.3394	0.1400	Tolga

Suite page suivante...

Expérience	Th	Area	Val oF1	Test oF1	Test fgd_oF1	Dice	IoU	P	R	Prénom
Autres architectures										
segformer_b5_type2	0.75	100	0.2077	0.1954	0.0853	0.1910	0.1624	0.1082	0.1097	Tolga/Ozan
segformer_b3_type2	0.75	100	0.2163	0.1880	0.0407	0.1698	0.1494	0.0965	0.0609	Tolga/Ozan
cmsegnet_type0	0.75	100	0.4619	0.4605	0.0186	0.4788	0.4663	0.8624	**0.5354**	Mohamed
cmsegnet_type2	0.5	100	0.5088	0.4917	0.0116	0.4761	0.4687	**0.9073**	0.5010	Mohamed

6 Conclusion

Ce travail a exploré la détection et la segmentation automatique de falsifications par copie-déplacement dans des images biomédicales. Grâce à un pipeline complet d'entraînement, validation et test, nous avons comparé plusieurs architectures modernes, révélant que les modèles auto-supervisés basés sur les transformeurs (DINOv2, DINOv3) offrent les meilleures performances (oF1 $\approx$ 79%, Dice $\approx$ 82%), tandis que les architectures convolutionnelles classiques et CMSeg-Net restent moins efficaces. Les études d'ablation ont montré l'importance de l'augmentation de données, du choix des fonctions de perte et du post-traitement (binarisation des masques et suppression des petites régions). Ces résultats confirment le potentiel des transformeurs et de l'apprentissage auto-supervisé pour cette tâche et ouvrent des perspectives d'amélioration par des modèles plus larges, l'apprentissage semi-supervisé ou des stratégies d'ensemble.

ANNEXE

7 Plannings Initial et Final

7.1 Planning Initial (soumis le 07/01/2026)

Le planning initial était structuré en 5 phases du 7 janvier au 10 février 2026, avec une approche progressive testant d'abord des modèles réduits avant les versions larges.

Phase	Dates	Modèles	Responsables	Actions
1	07/01–08/01	SegFormer-B0, DINOv2-Small, ConvNeXt-Tiny, SAM2-Small	Tous	Baseline : avec/sans augmentation
2	09/01	ConvNeXt-Tiny	Aymane	Test Focal Loss
3	10/01–11/01	Modèles larges (B2/B4/B5, Large)	Tous	Scale-up avec ajustements
4	12/01–14/01	Ajustements	Tous	Learning rate, post-processing
5	15/01	Soumission	Tous	Génération fichiers Kaggle

TABLE 5 – Planning d'entraînement initial

Modèles prévus : SegFormer (B0–B5), DINOv2 (Small/Large), ConvNeXt (Tiny/Base), SAM2 (Small/Large), SAFIRE.
Stratégie d'expérimentation :

1. Entraînement initial (tiny/base) avec/sans augmentation
2. Test fonctions de perte (BCE+Dice vs Focal+Dice)
3. Scale-up vers modèles larges avec fine-tuning learning rate
4. Sélection finale avec tests de confirmation

Entraînement progressif : Warm-up (époques 1-5, backbone gelé) puis fine-tuning complet.
Métriques suivies : train/val loss, Dice, IoU, oF1, pixel F1, precision, recall, learning rates.

Dates	Livrables Post-Kaggle
22/01–29/01	Optimisation du meilleur modèle
02/02–04/02	Rédaction et finalisation du rapport
08/02	Soumission code + rapport sur Moodle, préparation slides
09/02	Répétition de la présentation
10/02	Soutenance orale finale

TABLE 6 – Calendrier post-Kaggle

7.2 Planning Final Réellement Suivi

Date	Modèle	Configuration
08-15/01	**Entraînements pour la compétition Kaggle**	
16/01	U-Net+ResNet50	Type 0 (baseline)
17/01	U-Net+ResNet50	Type 1
18/01	U-Net+ResNet50	Type 2
19/01	U-Net+ResNet50	Type 2 + variantes paramètres
20/01	DINOv2+U-Net	Type 0
21/01	DINOv2+U-Net	Type 1
22/01	DINOv2+U-Net	Type 2
23/01	DINOv2+U-Net	Type 2 + decay modifié
24/01	DINOv2+U-Net	Type 2 + LR modifié
25/01	DINOv2+U-Net	Type 2 + Focal Loss
26/01	Segformer	B0
27/01	Segformer	B5
28/01	SegNet	Type 0
29/01	SegNet	Type 2 + config GitHub
30/01	DINOv3+U-Net	Type 0
31/01	DINOv3+U-Net	Type 1
01/02	DINOv3+U-Net	Type 2
02/02	DINOv3+U-Net	Type 2 + decay modifié
03/02	DINOv3+U-Net	Type 2 + UperNet
04/02	SAM	Type 0
05/02	SAM	Type 1
06/02	SAM	Type 2
07/02	SAM	Type 2 + decay modifié
08/02	ConvNeXt	Type 0
09/02	ConvNeXt	Type 1
10/02	ConvNeXt	Type 2
06-07/02	**Rédaction et finalisation du rapport**	
08/02	**Préparation slides présentation**	

TABLE 7 – Planning final réellement suivi (08/01/2026 – 08/02/2026)

7.3 Répartition du Travail entre les Membres

Membre	**Pourcentage**	**Rôle**	**Tâches Réalisées**
Tolga YILDIZ	30%	Monitoring, ajustement hyperparamètres, création de la pipeline et création du beamer, rédaction du rapport	U-Net+ResNet50, SegFormer, amélioration baseline DINOv2
Mohamed MOUDJAHED	30%	Chef de projet, ajustement hyperparamètres, création de la pipeline et création du beamer, rédaction du rapport	SAM2, DINOv3, SegNet, expérimentations DINOv2
Aymane ZENNOUHI	10%		ConvNeXt + U-Net, submission Kaggle
Ozan GUNES	30%	Monitoring, ajustement hyperparamètres, création de la pipeline et création du beamer, rédaction du rapport	DINOv2 baseline + amélioration, SegFormer
TOTAL	**100%**		

TABLE 8 – Répartition du travail entre les membres du groupe

Remarque : Cette répartition reflète la contribution réelle de chaque membre au projet, incluant l'implémentation des modèles, les expérimentations, les analyses, la rédaction du rapport,

et la préparation de la présentation.

8 Checklist de Complétion du Projet

Oui / Non	Exigence	Commentaires
Oui	Avez-vous relu votre rapport ?	
Oui	Avez-vous présenté l'objectif global de votre travail ?	
Oui	Avez-vous présenté les principes de toutes les méthodes/algorithmes utilisés ?	
Oui	Avez-vous cité correctement les références aux méthodes qui ne sont pas les vôtres ?	Ils sont présents dans Références
Oui	Avez-vous inclus tous les détails pour reproduire les résultats expérimentaux ?	Les détails sont dans le README.md
Oui	Avez-vous fourni des courbes, résultats numériques et barres d'erreur ?	On n'a pas relancé plusieurs fois le meme entraînement par manque de ressources et de temps
Oui	Avez-vous commenté et interprété les différents résultats ?	
Oui	Avez-vous inclus toutes les données, code et instructions pour reproduire les résultats ?	
Oui	Avez-vous organisé le code de manière unifiée ?	Présence d'une pipeline
Oui	Les résultats des différentes expériences sont-ils comparables ?	Nous avons à chaque fois changé qu'un hyperparamètre
Oui	Avez-vous suffisamment commenté votre code ?	
Oui	Avez-vous ajouté une documentation complète sur le code ?	
Oui	Avez-vous fourni les plannings provisoire et final dans le rapport ?	
Oui	Avez-vous décrit précisément les algorithmes non vus en cours ?	
Oui	Avez-vous fourni les pourcentages de charge de travail entre les membres ?	
Oui	Avez-vous envoyé le travail à temps ?	

9 Exemple de visualisation de falsification

La figure 7 présente des exemples de visualisation des prédictions de falsification pour chacun des modèles étudiés. Les couleurs des superpositions correspondent aux catégories suivantes :

— **Vert** : TP (Vrai Positif) – zone falsifiée correctement détectée

— **Rouge** : FP (Faux Positif) – zone non falsifiée détectée à tort comme falsifiée

— **Jaune** : FN (Faux Négatif) – zone falsifiée non détectée

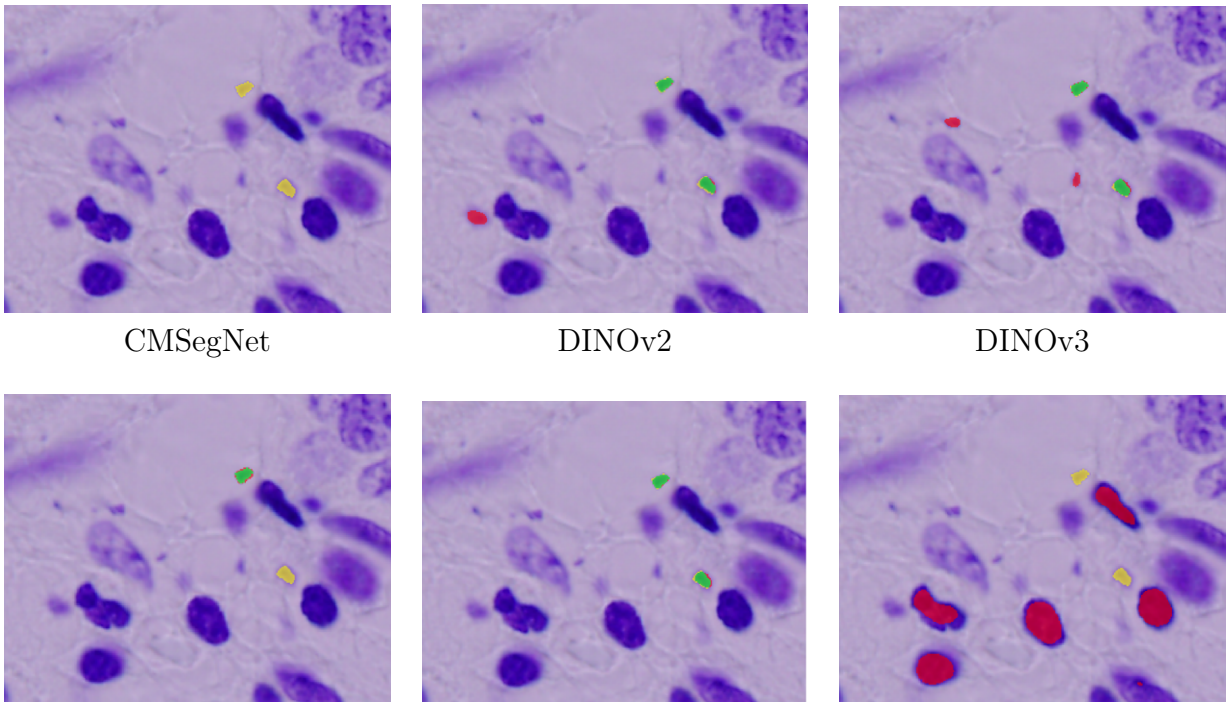

FIGURE 7 – Exemples de visualisation des prédictions de falsification pour chaque modèle.

10 Participation à une compétition Kaggle

Dans le cadre de ce travail, nous avons participé à la compétition Kaggle *Recod.ai/LUC – Scientific Image Forgery Detection* afin d'évaluer notre pipeline, bien que le temps limité dont nous disposions ne nous ait pas permis d'effectuer davantage d'entraînements.

FIGURE 8 – Capture d'écran d'une soumission effectuée sur la plateforme Kaggle

11 Drive avec les modèles pth

Google Drive : https://drive.google.com/drive/folders/1mkST-0apNJf8yQ7Pm7903QvWIXv6yscF

Références

[1] João Phillipe Cardenuto, Daniel Moreira, Anderson Rocha, Sohier Dane, Addison Howard, and Ashley Oldacre. Recod.ai/luc - scientific image forgery detection. Kaggle competition, 2025. Accessed 2026-02-07.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[4] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. *arXiv preprint arXiv :1612.03144*, 2016.

[5] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, 2022.

[6] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2 : Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024. arXiv preprint arXiv :2304.07193.

[7] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2 : Segment anything in images and videos. *arXiv preprint arXiv :2408.00714*, 2024.

[8] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Michael Maire, Piotr Dollár, and Kaiming He. Hiera : A hierarchical vision transformer without the bells-and-whistles. In *International Conference on Machine Learning (ICML)*, pages 29627–29645. PMLR, 2023.

[9] Hao-Chiang Shao, Yu-Ren Liao, Ting-Yu Tseng, Yan-Lin Chuo, and Fong-Yi Lin. Copy-move detection in optical microscopy : A segmentation network and a dataset. *IEEE Signal Processing Letters*, 2025. arXiv preprint arXiv :2412.10258.

[10] Oriane Siméoni et al. DINOv3. *arXiv preprint arXiv :2508.10104*, 2025.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[12] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer : Simple and efficient design for semantic segmentation with transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Recod.ai/LUC – Scientific Image Forgery Detection, Kaggle Competition.